DEO

IV

Rad sa eksternim podacima

U OVOM DELU:

POGLAVLJE

14

Integracija sa drugim aplikacijama

Iako Excel zaista predstavlja jednu univerzalnu aplikaciju, ne može se reći da je on apsolutno najbolji alat za sve vrste poslova, kao što neki bezuspešno pokušavaju da dokažu. Srećom, na raspolaganju imate mogućnost da u bilo koji Excel dokument "ugnjezdite" (embed), ili uz pomoć linkova povežete, dokumente kreirane u nekim drugim aplikacijama. Ovakvi dokumenti, koji su kreirani upotrebom više različitih aplikacija, nazivaju se složenim (compound) dokumentima.

Pored toga, Excel vam pruža mogućnost da programskim putem ostvarite kontrolu nad drugim aplikacijama. To je naročito korisno u situacijama kada je potrebno automatizovati neki proces, koji zahteva upotrebu većeg broja aplikacija. Sam čin uspostavljanja programske kontrole nad nekom aplikacijom, iz neke druge aplikacije, naziva se automatizacijom (automation).

Ovo poglavlje sadrži opšti prikaz tehnologija vezanih za kreiranje složenih dokumenata i automatizaciju, kao i primere praktične upotrebe svake od ovih tehnika.

Uvod u automatizaciju Officea

Tokom proteklih godina, Microsoft je, za označavanje tehologija i tehnika koje će biti obrađene u ovom odeljku, koristio veliki broj različitih termina. Kada budete proučavali druge knjige i novinske članke na ovu temu, susretaćete se sa sledećim terminima, u zavisnosti od datuma njihovog objavljivanja.

Object Linking and Embedding (OLE) Mada je termin "povezivanje i gnežđenje objekata" oduvek bio neraskidivo vezan za koncept složenih dokumenata, akronim OLE se ponekad koristio i za označavanje mnogo šireg spektra različitih tehnologija.

ActiveX Taman kad su se ljudi navikli na upotrebu termina OLE, Microsoft je za označavanje iste ove tehnologije najednom počeo da koristi termin ActiveX. Ova zamena termina je učinjena namerno, uz pretpostavku da će novi termin mnogo bolje odražavati proširenu funkcionalnost koju su omogućavale oslanjajuće tehnologije, sa posebnim akcentom na mogućnosti koje je ActiveX pružao po pitanju postavljanja dinamičkih sadržaja na web stranice. Može se, dakle, reći da ActiveX predstavlja više marketinški termin.

Component Object Model (COM) COM predstavlja široko prihvaćeni standard arhitekture softverskih komponenti i na njemu se zasnivaju sve ostale tehnologije koje će biti opisane u ovom poglavlju.

COM+ Ovo je poboljšana verzija COM-a, koja je po prvi predstavljena u operativnom sistemu Windows 2000.

Automatizacija Termin "automatizacija" (automation) se odnosi na koncept kontrolisanja jedne aplikacije, iz neke druge aplikacije.

NAPOMENA Microsoft poseduje i jednu noviju softversku arhitekturu, pod nazivom .NET (izgovara se kao "dot-net"). Radi se o potpuno novoj arhitekturi, koja nije direktno kompatibilna sa COM tehnologijama (doduše, ove dve tehnologije mogu ko-egzistirati uz pomoć jedne vrste sloja za prevođenje, koji nosi naziv "interop assembly"). Kako se softverski paket Microsoft Office u potpunosti zasniva na COM-u, to ćemo se u ovom poglavlju baviti isključivo COM tehnologijama.

Imajte na umu da se navedeni termini odnose na skup tehnologija koje u sebi obuhvataju više, možda čak i mnogo više od malog podskupa funkcionalnosti obrađenih u ovoj knjizi.

Moram da vam saopštim da sam ovde pokušao da stvari što više pojednostavim i izložene informacije svedem na neku razumnu meru. Ukoliko spadate u kategoriju ljudi koja uživa da kopa po detaljima, proučavanjem ove tehnologije moći ćete da ispunite nebrojene sate svog slobodnog vremena. Čim malo dublje zagazite u ovu oblast, stvari vrlo brzo postaju prilično komplikovane. Čitave knjige su posvećene ogromnoj složenosti COM tehnologije.

Ali, zbog čega se za označavanje ove tehnologije koristi toliko različitih termina? Da bih vam to objasnio, dozvolite mi kratku digresiju vezanu za istoriju njenog nastanka. Krajem 1980-ih, ukoliko ste želeli da podatke iz jedne aplikacije upotrebite u nekoj drugoj aplikaciji, niste imali baš mnogo izbora. Jedini mehanizam koji je prosečan korisnik imao na raspolaganju, bio je clipboard. Drugim rečima, tekst iz neke aplikacije ste mogli da isečete ili iskopirate u clipboard, te da ga zatim zalepite u drugu aplikaciju. Za svakodnevne potrebe je to bilo sasvim dovoljno, ali korisnici su zahtevali mnogo više od toga. Ukoliko bi, na primer, došlo do izmene podataka u jednom dokumentu, sve zavisne dokumente biste morali ručno da ažurirate. Jasno je, dakle, da je obezbeđenje stalne ažurnosti srodnih dokumenata, u to vreme predstavljalo veoma naporan zadatak.

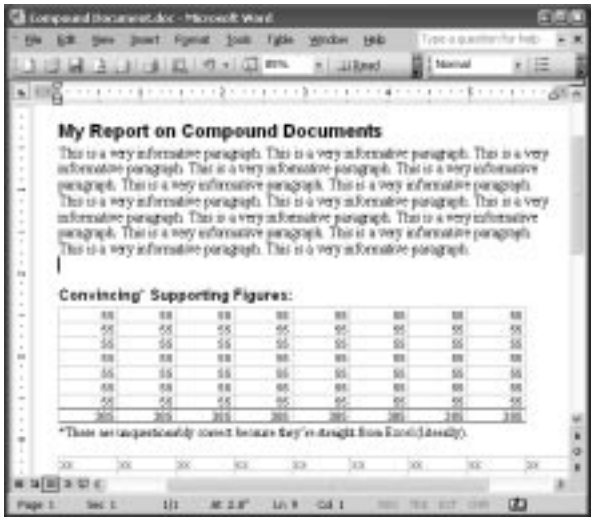
Radi rešenja problema ažuriranja identičnih informacija na svim zavisnim dokumentima u slučaju promene tih informacija na izvornom dokumentu, Microsoft je kreirao takozvanu dinamičku razmenu podataka (dynamic data exchange - DDE), kao i početnu verziju tehnologije povezivanja i gnežđenja objekata (OLE). Mada se uz pomoć DDE pomenuti problem može rešiti, ova tehnologija uopšte nije bila jednostavna za upotrebu.

Originalna verzija OLE tehnologije značajna je prvenstveno po tome što je Microsoft u njoj po prvi put predstavio koncept složenih dokumenata. Kao što je već rečeno, složeni dokument je onaj dokument koji je kreiran upotrebom više različitih aplikacija. To je praktično značilo da ste mogli kreirati Word dokument, koji bi u sebi sadržao neki grafikon ili radni list iz Excela, poput dokumenta koji je prikazan na slici 14.1. Ovo je bila zaista fantastična ideja, ali, na žalost, malo ispred svog vremena. Naime, većina PC računara iz ovog perioda se, u najmanju ruku, teško snalazila sa složenim dokumentima. Sećam se da su se neke od najvećih demonstracija moje strpljivosti odigrale upravo pri pokušaju kreiranja što impresivnijih složenih dokumenata.

Lavina koja je pokrenuta početnom verzijom OLE tehnologije, neminovno je dovela do pojave njene znatno poboljšane verzije. Međutim, ovog puta je Microsoft potpuno odbacio sve reference tipa "Object Linking and Embedding (OLE)" i novu tehnologiju jednostavno nazvao OLE, dajući tako svima do znanja da ona podrazumeva mnogo više od jednostavnog kreiranja složenih dokumenata. OLE je, naime, predstavljao fundamentalnu promenu u načinu razmišljanja. Microsoftovi inženjeri koji su radili na projektovanju originalne verzije OLE-a sada su počeli da na delove složenog dokumenta gledaju kao na softverske komponente. Posledica ovakve promene vizure bila je da su projektanti započeli sa definisanjem načina na koji će ove softverske komponente međusobno "razgovarati" i otkrivati servise koje svaka od njih nudi. U tu svrhu je Microsoft razvio standard pod nazivom Component Object Model (COM). Druga verzija OLE-a je, dakle, izgrađena upravo na COM temeljima.

Sredinom 90-ih godina, Microsoft je lansirao termin ActiveX i (još jednom) potpuno zbunio sve svoje korisnike. Sve je najedanput postalo ActiveX - ActiveX kontrole, ActiveX dokumenta, ActiveX skriptovanje. Mislim da je otprilike u to vreme započeo i razvoj Xbox-a. Ne bi me čudilo ako je i ovaj projekat u početku nosio tajni naziv ActiveXbox!

U čemu je, dakle, razlika između ActiveX kontrola i OLE kontrola? Ni u čemu. Koja je onda svrha ove promene termina? Radi se o čisto marketinškom potezu. Ah da, umalo da zaboravim - sa pojavom ActiveX-a, termin OLE je ponovo počeo da se upotrebljava isključivo za označavanje tehnologije povezivanja i gnežđenja dokumenata, a ne za mnogo širi spektar tehnologija kao što je to bio slučaj u prethodnom periodu. U to vreme su čak i Homera Simpsona citirali samo sa: "DOOUGH!"



SLIKA 14.1
Koncept složenih dokumenata iznedrio je jedan potpuno nov način razmišljanja.

Imajući sve ovo u vidu, osnovna stvar koju bi trebalo da upamtite iz ovog odeljka jeste da se, bez obzira na to koji je tehnički ili marketinški termin upotrebljen, koncept o kome ovde govorimo prvenstveno odnosi na jedan od sledeća dva procesa: Prvo, kreiranje i manipulacija složenim dokumentima, i drugo, automatizacija jedne aplikacije iz neke druge aplikacije.

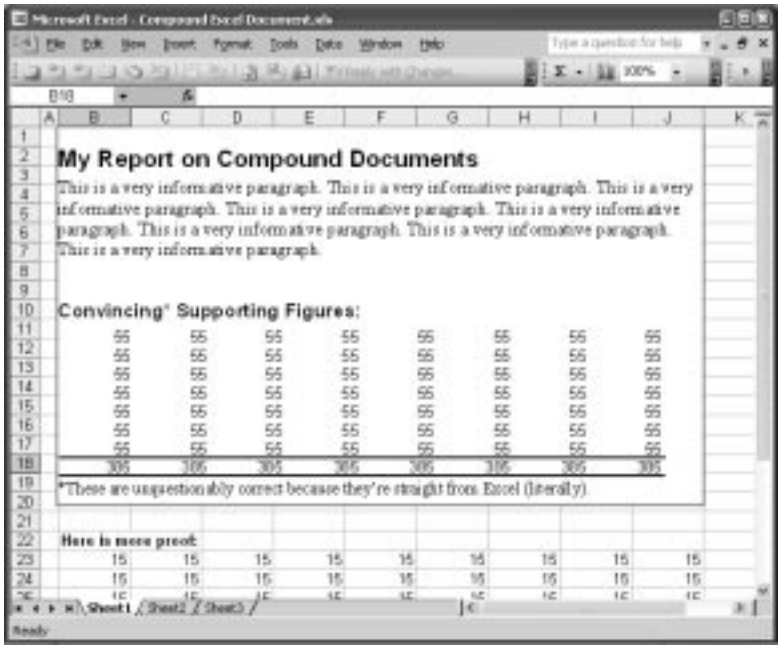
Opširnije o složenim dokumentima

Kao što sam pomenuo u prethodnom odeljku, složeni dokumenti su kreirani upotrebom više različitih aplikacija, pri čemu se svaki od njih u svojoj matičnoj (host) aplikaciji pojavljuje kao jedan objedinjeni dokument. Prema tome, bilo koji složeni dokument možete, unutar matične aplikacije, pregledati u svoj njegovoj celovitosti.

Komponente nekog složenog dokumenta mogu biti uskladištene na dva različita načina. Prvi način podrazumeva upotrebu linkova (veza). Kada neki složeni dokument kreirate uz pomoć linkova, onda će fajl složenog dokumenta sadržati u sebi samo linkove, dok će se konkretne komponente čuvati u posebnom fajlu.

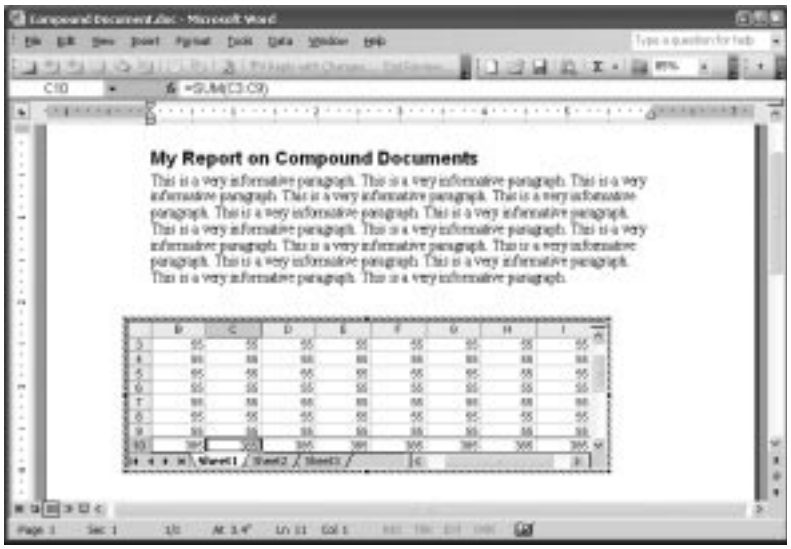
Drugi način se sastoji u gnežđenju komponenata direktno u fajl složenog dokumenta. Upravo zbog toga se ova tehnologija naziva povezivanjem i gnežđenjem objekata (OLE) - terminom koji neduhovito, ali prilično tačno opisuje suštinu složenih dokumenata.

Složeni dokument se može sastojati od velikog broja komponenata. Ukoliko imate avanturistički duh, ili vam je prosto dosadno, možete praktično isprobati svaku kombinaciju koja vam padne na pamet. Primera radi, na slici 14.2 prikazan je Excelov radni list, ugnežđen unutar jednog Word dokumenta, koji je, pak, smešten unutar drugog radnog lista u Excelu. Ovaj trik mi je pokazao muž sestre od tetke najboljeg prijatelja mog pašenoga.



SLIKA 14.2 *Gnežđenje Excela u Word dokument koji je ugnežđen u Excel? Ovakvim gnežđenjem različitih komponenata jedne u drugu možete kreirati složene dokumente od bilo koje kombinacije komponenata koja vam padne na pamet.*

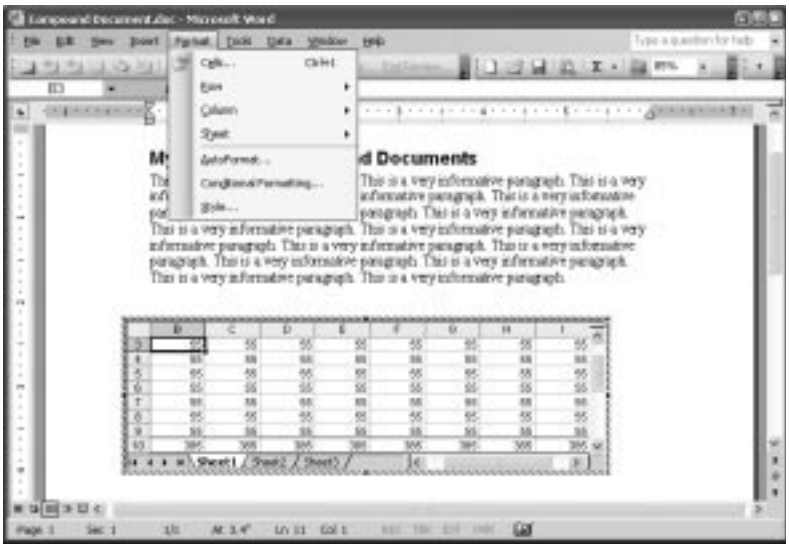
U zavisnosti od toga da li je neka komponenta ugnežđena ili linkovana, nad njom ćete moći da izvršavate različite aktivnosti. Primera radi, nad ugnežđenim komponentama možete izvršiti tzv. editovanje na licu mesta (in-place), kao što je to ilustrovano na slici 14.3, ili ih u punom obimu otvoriti u njihovim matičnim aplikacijama i tek potom ih editovati.



SLIKA 14.3 *Ugnežđene objekte možete editovati na licu mesta.*

Po mom mišljenju, editovanje na licu mesta predstavlja zaista impresivnu razvojnu opciju. Pogledajte, na primer, šta se dešava na slici 14.4, pri pokušaju editovanja Excelovog radnog lista

koji je ugnežđen u jedan Word dokument. Naime, Word će automatski poprimiti oblik Excelovog radnog okruženja. Pod-meni koji je prikazan na slici 14.4 predstavlja deo Excelovog menija, iako ja ovde uopšte nisam izašao iz Word okruženja, što se može zaključiti i po Word ikonici u gornjem levom uglu prozora.



SLIKA 14.4 *Takozvano "in-place" editovanje ugnežđenih komponenata u nekom složenom dokumentu predstavlja zaista impresivnu mogućnost.*

Linkovane komponente se takođe mogu editovati. Ipak, za razliku od ugnežđenih komponenata, na linkovane komponente se ne može primeniti editovanje na licu mesta. Osnovna prednost upotrebe linkovanih komponenti je u tome što možete kreirati "živi" dokument, koji je u potpunosti svestan svih linkovanih komponenti koje su u njemu smeštene. Kada neka linkovana komponenta pretrpi izvesne izmene, u složenom dokumentu će te izmene biti automatski ažurirane, bez ikakve intervencije sa vaše strane.

Izrada složenih dokumenata programskim putem

Sa tačke gledišta projektanta, zar ne bi bilo sjajno malo dublje zaroniti u funkcionalnost koju nudi tehnologija povezivanja i gnežđenja objekata? Verovali ili ne, iznenađujuće je jednostavno programskim putem ugnjezditi ili kreirati link ka nekom dokumentu, unutar nekog drugog dokumenta. Excelov objektni model sadrži u sebi dva objekta koja se odnose na povezivanje i gnežđenje objekata: OLEObject and OLEObjects. Pri tom, OLEObjects predstavlja kolekcijski objekat, koji u sebi može sadržati veći broj OLEObject objekata. Poput drugih kolekcijskih objekata sa kojima ste se do sada upoznali, OLEObjects sadrži metod Add, pomoću kojeg možete dodati nove OLEObject objekte na radni list. Iako metod Add sadrži obilje parametara, svi su oni, na neki način, opcionog karaktera, i obično je potrebno definisati samo nekoliko njih. Kažem: "na neki način", jer je neophodno definisati bar jedan od parametara ClassType ili FileName. Evo kako glasi sintaksa Add metoda:

```
SomeWorksheetObject.OLEObjects.Add [ClassType], [FileName], [Link],
[DisplayAsIcon], [IconFileName], [IconIndex], [IconLabel], [Left],
[Top], [Width], [Height]
```

Svi parametri metoda Add ukratko su objašnjeni na sledećoj listi.

ClassType Programski identifikator objekta koji treba kreirati. Ako definišete parametar ClassType, onda će parametri FileName i Link biti ignorisani. Pri tom je neophodno definisati bar jedan od parametara ClassType ili FileName. Parametar ClassType treba upotrebiti u slučajevima kada novi OLE objekat želite da dodate na radni list koji se ne zasniva na nekom od postojećih dokumenata. Primera radi, ako ClassType parametru dodelite vrednost "Word.Document", u tekući radni list će biti ugnežđen jedan prazan Word dokument.

FileName Naziv fajla na kome će se novi OLE objekat zasnivati, odnosno sa kojim će biti linkovan. Neophodno je definisati bar jedan od parametara ClassType ili FileName. Prametar FileName upotrebićete u slučajevima kada želite da ugnjezdite ili linkujete neki od postojećih dokumenata.

Link Parametar Link se upotrebljava isključivo u kombinaciji sa FileName parametrom. Ukoliko želite da novi dokument ugnjezdite u radni list, onda bi parametru Link trebalo dodeliti vrednost "false". U suprotnom, ako želite da novi OLE objekat bude linkovan sa izvornim dokumentom, parametar Link bi trebalo da ima vrednost "true". Njegova podrazumevana (default) vrednost je "false".

DisplayAsIcon Ovom parametru treba dodeliti vrednost "true", ukoliko želite da novi OLE objekat bude na radnom listu prikazan u obliku ikonice; u suprotnom, ako mu dodelite vrednost "false", OLE objekat će biti prikazan u normalnom obliku. Ukoliko mu dodelite vrednost "true", za definisanje ikonice možete upotrebiti parametre IconFileName i IconIndex. Podrazumevana vrednost ovog parametra je "false".

IconFileName Ovaj parametar predstavlja naziv fajla koji u sebi sadrži ikonicu koju treba prikazati. Upotrebljava se jedino u slučajevima kada je parametru DisplayAsIcon dodeljena vrednost "true".

IconIndex Indeksni broj željene ikonice unutar fajla sa ikonicama. Ovaj parametar se upotrebljava jedino pod uslovom da DisplayAsIcon ima vrednost "true" i da parametar IconFileName ukazuje na validni fajl sa ikonicama.

IconLabel Ovaj parametar bi trebalo da predstavlja tekstualni string, koji će biti prikazan ispod ikonice. Upotrebljava se jedino u slučaju da je parametru DisplayAsIcon dodeljena vrednost "true". Po podrazumevanoj vrednosti, ispod ikonice neće biti prikazan nikakav tekst.

Left Razdaljina, u tačkama, od leve ivice radnog lista do leve ivice objekta. Po podrazumevanoj vrednosti, ovaj parametar će imati istu vrednost kao i Left svojstvo aktivne ćelije (ActiveCell.Left).

Top Razdaljina, u tačkama, od gornje ivice radnog lista do gornje ivice objekta. Po podrazumevanoj vrednosti, ovaj parametar će imati istu vrednost kao i Top svojstvo aktivne ćelije (ActiveCell.Top).

Width Početna širina objekta, izražena u tačkama (points).

Height Početna visina objekta u tačkama.

SAVET Nemate pojma kako da procenite potrebnu udaljenost u tačkama? Ne paničite - nemam ni ja. Pod pretpostavkom da ste američki državljanin, ili ste na drugi način upoznati sa našim jedinstvenim načinom merenja razdaljine, mnogo komfornije ćete se osećati ukoliko za definisanje razdaljina upotrebite metod Application.InchesToPoints. Primera radi, ako želite da OLE objekat bude postavljen na pola inča od gornje ivice radnog lista, možete upotrebiti izjavu poput sledeće:

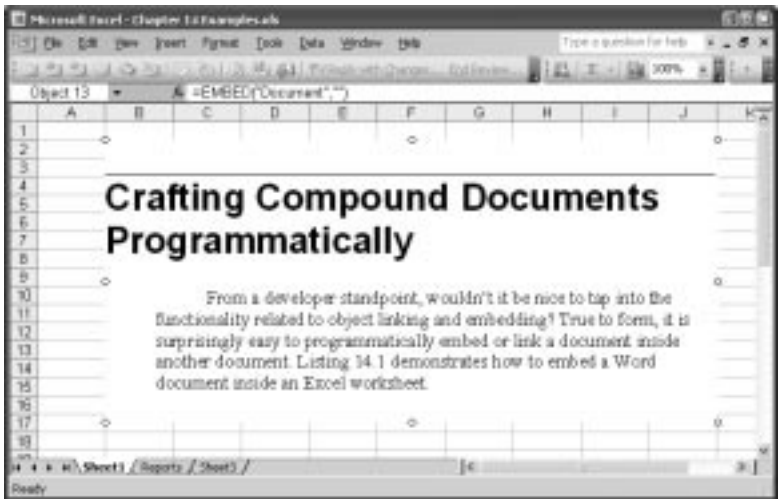
```
MyOLEObject.Top = Application.InchesToPoints(0.50).
```

U listingu 14.1 ilustrovan je jedan od načina na koji, programskim putem, neki Word dokument možete ugnjezditi u Excelov radni list, uz pomoć Add metoda OLEObjects objekta.

Listing 14.1: Kreiranje složenih dokumenata programskim putem, upotrebom OLEObjects objekta

```
Sub CreateCompoundDocument()
    Dim rg As Range
    Dim obj As OLEObject

    ' Definisanje opsega koji ce ukazivati
    ' na gornji levi ugao OLEObject objekta
    Set rg = ThisWorkbook.Worksheets(1).Cells(2, 2)

    ' Insertovanje OLEObject objekta
    Set obj = InsertObject(rg, "C:\testdoc.doc", False)

    ' Obavestavanje korisnika da je novi objekat insertovan (ili nije)
    If Not obj Is Nothing Then
        Debug.Print "Object inserted."
    Else
        Debug.Print "Sorry - the object could not be inserted."
    End

    ' Ciscenje
    Set obj = Nothing
    Set rg = Nothing
End Sub

Function InsertObject(rgTopLeft As Range, sFile As String, _
    bLink As Boolean) As OLEObject

    Dim obj As OLEObject

    On Error GoTo ErrHandler

    ' Insertovanje objekta
    Set obj = rgTopLeft.Parent.OLEObjects.Add( _
        Filename:=sFile, _
        Link:=bLink)

    ' Ove parametre nemojte definisati unutar Add metoda (gore),
    ' jer ce to dovesti do pojave greske.
    obj.Top = rgTopLeft.Top
    obj.Left = rgTopLeft.Left

    ' Vraca referencu ka insertovanom OLEObject objektu
    Set InsertObject = obj
    Exit Function

ErrHandler:
    ' Tartar sos! Doslo je do pojave greske.
```

```
        Debug.Print Err.Description
        Set InsertObject = Nothing
    End Function
```



SLIKA 14.5 *Kreiranje složenih dokumenata programskim putem je pravi mačiji kašalj, ukoliko u tu svrhu upotrebite OLEObjects objekat.*

Cilj ovog listinga se sastoji u gnežđenju Word dokumenta pod nazivom "C:\testdoc.doc" (istoimeni fajl ćete i vi morati da imate na istoj toj lokaciji, ako želite da praktično isprobate ovaj listing), na prvi po redu radni list u tekućoj radnoj knjizi, pri čemu će gornji levi ugao ugnežđenog Word dokumenta biti smešten u ćeliju B2. Da biste to mogli postići, kreirao sam funkciju pod nazivom InsertObject, koja u stvari vrši gnežđenje, odnosno linkovanje dokumenta, unutar radnog lista na kome je smešten pomenuti opseg. Procedura pod nazivom CreateCompoundDocument definiše odgovarajuće parametre, koji su neophodni radi obavljanja "probne vožnje" InsertObject funkcije. Rezultati ove probne vožnje prikazani su na slici 14.5.

Jedna od interesantnih stvari u vezi listinga 14.1 odnosi se na činjenicu da uopšte ne morate navoditi naziv aplikacije ili klase, pomoću koje je kreiran dokument koji insertujete. Ukoliko, na primer, funkciji InsertObject prosledite neki PowerPoint fajl, ona će besprekorno izvršiti ovu vašu naredbu i na radni list ugnjezditi željeni PowerPoint dokument (videti sliku 14.6). U stvari, za besprekorno izvršavanje pomenute naredbe zaslužan je Add metod OLEObjects objekta, jer je upravo ovaj metod pozvan od strane InsertObject funkcije.

OLE je sjajan; automatizacija je još bolja

Linkovanje i gnežđenje objekata predstavlja zaista sjajnu mogućnost, ali šta ako je potrebno da na linkovanom ili ugnežđenom dokumentu izvršite neku akciju? Ili, šta ako je neophodno da u programskom kôdu upotrebite funkcionalnost koju pruža neka druga aplikacija? Za izvršavanje ove vrste zadataka biće neophodno da izađete iz okvira linkovanja i gnežđenja objekata i uđete u oblast koja se generalno naziva automatizacijom. Pod automatizacijom se podrazumeva proces pristupa objektnom modelu koji je pridružen nekoj drugoj aplikaciji ili softverskoj komponenti.

SLIKA 14.6 *Zašto se zadovoljiti samo Word dokumentima? Na radne listove možete ugnjezditi i dokumente koji su kreirani u nekim drugim aplikacijama, uključujući i PowerPoint.*

Kada jednom naučite mehanizme za pristup biblioteci klasa (class library) neke druge aplikacije, bićete, što se kaže, na konju. Naime, nakon toga ćete sve biblioteke klasa moći da upotrebljavate na isti način kao kada to činite iz njihovih matičnih aplikacija. Naravno, to podrazumeva da budete dobro upoznati sa bibliotekom klasa konkretne aplikacije. Isto kao što se malo toga može uraditi u Excelu bez dobrog poznavanja Excelovog objektnog modela, tako ćete i u Wordu teško šta postići dok u dovoljnoj meri ne ovladate Wordovim objektnim modelom.

Kako, dakle, da pristupite biblioteci klasa neke druge aplikacije? To prvenstveno zavisi od načina na koji želite da se spojite (bind) sa tom bibliotekom.

Spajanje sa nekom bibliotekom klasa

Jedan od koncepata koje je veoma važno dobro razumeti, jeste koncept takozvanog spajanja (binding). Spajanje se dešava u trenutku kada kontrolišuća aplikacija (ili klijent) spozna sve potrebne informacije o objektima, svojstvima i metodima koji su prisutni unutar objektnog modela kontrolisane aplikacije (ili servera).

Postoje dva tipa spajanja: rano spajanje i kasno spajanje. Rano (early) spajanje se dešava još u toku faze projektovanja i (obično) zahteva od projektanta da postavi odgovarajuće reference ka tipu biblioteke koja je pridružena datom serveru. Sa druge strane, kasno (late) spajanje se odigrava u fazi izvršavanja aplikacije (run-time).

U većini slučajeva bi trebalo da koristite rano spajanje. Rano spajanje nudi dve važne prednosti: prvo, ono pruža znatno brže performanse; drugo, ono vam omogućava upotrebu korisnih VBE karakteristika, kao što su Auto Syntax Check, Auto List Members i brz pristup kontekstno-senzitivnim help fajlovima na serveru. Međutim, rano spajanje ima i jedan nedostatak - ukoliko jednu ovakvu aplikaciju upotrebite na PC računaru na kome neophodna serverska aplikacija nije instalirana, onda vaša aplikacija neće funkcionisati i doći će do pojave greške čim pokušate da otvorite aplikacijski fajl. Razlog za ovo leži u činjenici da, odmah po otvaranju fajla, VBA automatski vrši proveru validnosti svih rano spojenih objekata.

Uključivanje opcije ranog spajanja vrši se postavljanjem reference ka tipu biblioteke željenog servera. Radi postavljanja pomenute reference, sa VBE menija odaberite komandu Tools➥References, nakon čega će se na ekranu otvoriti okvir za dijalog sličan onome koji je prikazan na slici 14.7.

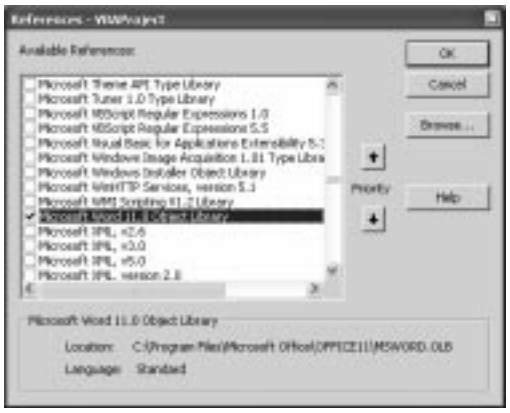
Da biste omogućili rano spajanje, pregledajte listu raspoloživih referenci, pa zatim označite polje za potvrdu pored odgovarajućeg tipa biblioteke. Na ovaj način bi trebalo da selektujete samo one tipove biblioteka koje zaista planirate da upotrebite. Na slici 14.8, ja sam postavio referencu ka Microsoft Wordu.

Treba imati na umu da rano spajanje ne podržavaju baš svi serveri, tako da ćete u tom slučaju morati da primenite kasno spajanje. Kasno spajanje je znatno sporije od ranog spajanja, zbog specijalnog procesa validacije, koji se u pozadini izvršava za svaku liniju programskog kôda u kojoj je određeni server upotrebljen.

Da se vratimo, dakle, na naše početno pitanje. Na koji način možete pristupiti biblioteci klasa neke eksterne aplikacije? U listingu 14.2 ilustrovana su dva moguća načina pristupa Wordovoj biblioteci klasa.

SLIKA 14.7
Uključenje ranog spajanja vrši se uz pomoć okvira za dijalog References.

SLIKA 14.8
Svaka od raspoloživih referenci koju selektujete, biće rano spojena.



Listing 14.2: Uporedni prikaz ranog i kasnog spajanja

```
Sub WordEarlyBound()
    Dim wd As Word.Application
    Dim doc As Word.Document

    ' Kreiranje nove instance Worda
    Set wd = New Word.Application

    ' Dodavanje novog dokumenta
    Set doc = wd.Documents.Add

    ' Snimanje i zatvaranje dokumenta
    doc.SaveAs "C:\testdoc1.doc"
    doc.Close

    ' Ciscenje
    Set doc = Nothing
    Set wd = Nothing
End Sub

Sub WordLateBound()
    Dim objWord As Object
    Dim objDoc As Object

    ' Kreiranje nove instance Worda
    Set objWord = CreateObject("Word.Application")

    ' Dodavanje novog dokumenta
    Set objDoc = objWord.Documents.Add

    ' Snimanje i zatvaranje dokumenta
    objDoc.SaveAs "C:\testdoc2.doc"
    objDoc.Close

    ' Ciscenje
    Set objDoc = Nothing
    Set objWord = Nothing
End Sub
```

Funkcionisanje procedure WordEarlyBound zasniva se na pretpostavci da ste prethodno već postavili referencu ka Microsoft Word Object Library, tako što ste ovu opciju selektovali preko okvira za dijalog koji je prikazan na slici 14.8. Da biste saznali da li je u programskom kôdu upotrebljena tehnika ranog spajanja, biće dovoljno da pogledate sekciju u kojoj je izvršeno deklarisanje varijabli. Postavljanjem reference ka biblioteci Microsoft Word Object Library, vi u stvari VBA upoznajete sa svim potrebnim informacijama u vezi Wordovog objektnog modela. Kao rezultat toga, varijable ćete moći da deklarišete preko specifičnog objekta koji je dodeljen svakoj varijabli, umesto da u tu svrhu upotrebljavate generički Object objekat.

Nasuprot tome, procedura pod nazivom WordLateBound uopšte ne zahteva postavljanje bilo kakvih referenci. Kao posledica toga, varijable ćete morati da deklarišete kao objekte tipa Object. Nadalje, s obzirom da VBA neće znati baš ništa o svojstvima i metodima koji su pridruženi svakoj varijabli, to od VBE editora nećete moći da očekujete nikakvu pomoć u obliku opcija Auto List Members ili Auto Syntax Checking. Pisanje kôda uz upotrebu kasno spojenih objekata pripada staroj školi VBA programiranja. Na taj način ćete biti prinuđeni ili da u glavi memorišete čitave biblioteke objekata, ili da puno vremena provedete u proučavanju help fajlova, kako biste saznali sve potrebne informacije u vezi upotrebe željenih svojstava, metoda i njima odgovarajućih parametara.

Praktičan primer: Automatizacija jedne PowerPoint prezentacije

Vreme je da najzad skinemo rukavice i pozabavimo se jednim praktičnim problemom, za koji bi automatizacija mogla predstavljati dobro rešenje. Za potrebe ovog primera, zamislite da ste finansijski analitičar jedne velike trgovinske kompanije - Bullseye Corporation. Jedan od vaših zadataka se sastoji u tome da svakog meseca pripremate prezentaciju za svog šefa, koji te podatke zatim prezentuje upravnom odboru. U tom mesečnom izveštaju treba da budu prikazane detaljne informacije o prodaji, kao što su uporedni prikazi rezultata prodaje u jednoj istoj prodavnici po mesecima, ukupan iznos ostvarenog prihoda po prodavnicama i drugi ključni indikatori uspešnosti prodaje.

Većina prezentacije se, dakle, sastoji od finansijskih podataka koje ste prethodno obradili u Excelu. S obzirom na to da je vaša kompanija pametno investirala novac u kupovinu Microsoft Analysis Services paketa koji omogućava znatno bržu analizu podataka u Excelu, potrebno vam je svega nekoliko minuta da prikupite sve podatke koji su neophodni za kreiranje mesečnog izveštaja.

Problem

Problem se sastoji u tome što prezentacija mora da bude kreirana u PowerPointu. Kao posledica toga, većinu vremena na njenoj izradi vi trošite na ručno prebacivanje podataka iz Excela u PowerPoint. Već ste izvršili određene eksperimente sa povezivanjem i gnežđenjem objekata, kao efikasnog načina za smanjenje ili potpunu eliminaciju napornog kopiranja i lepljenja podataka pri izvršavanju ovog zadatka.

Primera radi, pokušali ste da kreirate jednu jedinu radnu knjigu u Excelu i jednu PowerPoint prezentaciju. Pri tom, prezentacija u sebi sadrži link ka podacima iz Excela. Na kraju svakog meseca, sve što treba da uradite jeste da osvežite podatke u Excelu, nakon čega će ažuriranje PowerPoint prezentacije biti izvršeno potpuno automatski. Po potrebi, možete još jedino izmeniti izgled crtica (bullet points) u prezentaciji, kako one međusobno ne bi bile baš slične kao jaje jajetu. Prema teoriji, sve bi ovo trebalo da funkcioniše bez ikakvih problema. Ipak, suočili ste se sa dva problema u ovakvom pristupu, usled čega se ispostavilo da on nije toliko efikasan kao što se na početku ponadali.

Pre svega, vi ste, poput većine finansijskih analitičara, po samoj prirodi svog posla prinuđeni na svaštarenje. Neophodno je, naime, da čuvate kopije svih izveštaja koje ste kreirali, zajedno sa svim podacima koji su bili neophodni za njihovo kreiranje. Na taj način, ukoliko vam iko ikada postavi neko pitanje u vezi bilo kog izveštaja, moći ćete da se vratite unazad i pronađete tražene detalje. Prema tome, od maločas pomenutog pristupa, tipa "jedna radna knjiga/jedna prezentacija", nećete imati baš puno koristi, jer prilikom ažuriranja podataka za tekući mesec neminovno uništavate podatke iz prethodnog meseca. Osim toga, kompeksni sistem direktorijuma koji ste kreirali radi skladištenja fajlova istovremeno povećava rizik od pojave grešaka pri linkovanju fajlova. Svaki put kada neki fajl iskopirate na novu lokaciju, povećava se rizik od pojave grešaka bilo usled prekinutih linkova ili, što je još gore, usled upotrebe linkova koji su usmereni ka pogrešnim fajlovima. Primera radi, u prezentaciji za mesec oktobar može se pojaviti link ka Excelovoj radnoj knjizi za septembar.

Drugi problem se takođe odnosi na linkovanje. Naime, fajl prezentacije bi morao da bude prenosiv, kako bi vaš šef i ostali zainteresovani mogli da ga koriste bez potrebe za istovremenom distribucijom pratećeg Excel fajla. To praktično znači da u prezentaciju zaista morate ugnjezditi Excel fajl, a ne samo da kreirate link ka njemu. Naravno, ukoliko primenite tehniku gnežđenja fajla, nećete moći da koristite prednosti eliminisanja kopiranja i lepljenja, jer ćete svakog meseca morati iznova da kreirate novu prezentaciju.

Rešenje

Rešenje ovog problema sastoji se u primeni automatizacije u Excelu, čime će prezentacija svakog meseca biti automatski kreirana na osnovu podataka iz Excelove radne knjige. Uz pomoć VBA možete napisati programski kôd koji će:

- Kreirati instancu PowerPointa
- Kreirati praznu prezentaciju na osnovu definisanog uzorka
- Kreirati naslovni slajd prezentacije
- Kreirati po jedan slajd za svaki izveštaj koji ste kreirali u Excelu
- Sačuvati prezentaciju na lokaciji po vašem izboru

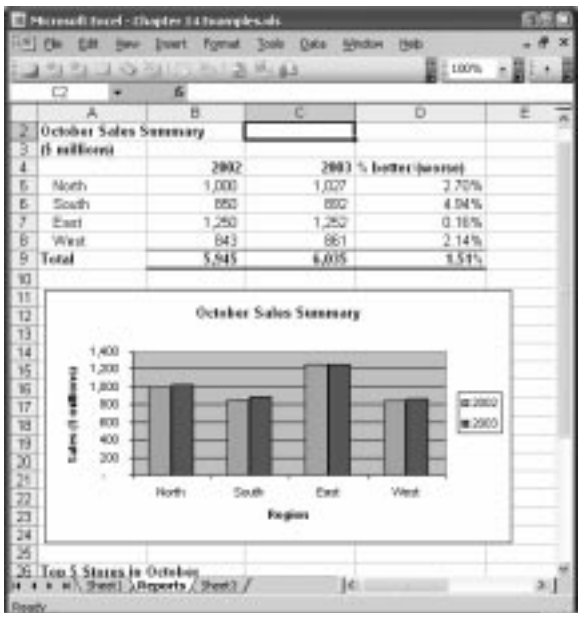
U idealnom slučaju, sve što ćete morati ručno da uradite, jeste da kreirate dodatni sadržaj prezentacije, koji ne potiče iz Excela.

Primer radne knjige

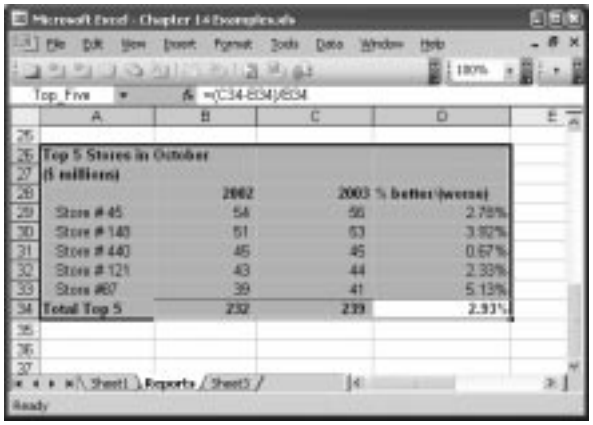
Za potrebe ovog primera kreirao sam tri izveštaja - svi su smešteni na radnom listu pod nazivom "Reports". Opseg u kome je smešten prvi izveštaj nosi naziv "Sales_Summary". Ovaj izveštaj je prikazan na slici 14.9, zajedno sa još jednim izveštajem, koji u stvari predstavlja grafikon kreiran na osnovu Sales Summary podataka.

Treći izveštaj predstavlja top listu pet prodavnica u kojima je ostvarena najveća prodaja. Imenovani opseg u kome je smešten ovaj izveštaj nosi naziv "Top_Five". Ovaj izveštaj je prikazan na slici 14.10.

SLIKA 14.9 *Izveštaj i grafikon pod nazivom Sales Summary*



SLIKA 14.10
Top lista pet najuspešnijih prodavnica



Programski kôd za implementaciju ovog rešenja prikazan je u listingu 14.3. U ovom listingu je upotrebljeno rano spajanje, radi uspostavljanja reference ka PowerPointu. Prema tome, ako želite da listing praktično isprobate, i vi ćete morati da postavite referencu ka PowerPointu tako što ćete u VBE odabrati komandu Tools➥References, pa zatim, na listi raspoloživih referenci, označiti polje za potvrdu pored opcije Microsoft PowerPoint 11.0 Object Library. Ukoliko koristite verziju Microsoft Officea stariju od Officea 2003, na raspolaganju ćete imati odgovarajuću verziju PowerPointa, kao na primer Microsoft PowerPoint 10.0 Object Library.

Listing 14.3: Automatizovano kreiranje PowerPoint prezentacije iz Excela

```
Option Explicit

Sub CreatePresentation()
    Dim ppt As PowerPoint.Application
    Dim pres As PowerPoint.Presentation
    Dim sSaveAs As String
    Dim ws As Worksheet
    Dim chrt As Chart
    Dim nSlide As Integer

    On Error GoTo ErrHandler

    Set ws = ThisWorkbook.Worksheets("Reports")

    ' Kreiranje nove innstance PowerPointa
    Set ppt = New PowerPoint.Application

    ' Kreiranje nove prezentacije
    Set pres = ppt.Presentations.Add
        pres.ApplyTemplate _
        "c:\program files\microsoft office\templates" & _
        "\presentation designs\maple.pot"
```

```
    With pres.Slides.Add(1, ppLayoutTitle)
        .Shapes(1).TextFrame.TextRange.Text = "October Sales Analysis"
        .Shapes(2).TextFrame.TextRange.Text = "11/5/2003"
    End With

    ' Kopiranje podataka
    CopyDataRange pres, ws.Range("Sales_Summary"), 2, 2
    CopyChart pres, ws.ChartObjects(1).Chart, 3, 1
    CopyDataRange pres, ws.Range("Top_Five"), 4, 2

    ' Snimanje i zatvaranje fajla prezentacije
    sSaveAs = GetSaveAsName("Save As")
    If sSaveAs <> "False" Then
        pres.SaveAs sSaveAs
    End If
    pres.Close

ExitPoint:
    Application.CutCopyMode = False
    Set chrt = Nothing
    Set ws = Nothing
    Set pres = Nothing
    Set ppt = Nothing
    Exit Sub
ErrHandler:
    MsgBox "Sorry the following error has occurred: " & _
        vbCrLf & vbCrLf & Err.Description, vbOKOnly
    Resume ExitPoint
End Sub

Private Sub CopyDataRange(pres As PowerPoint.Presentation, _
    rg As Range, nSlide As Integer, _
    dScaleFactor As Double)

    ' Kopiranje opsega u clipboard
    rg.Copy

    ' Dodavanje novog praznog slajda
    pres.Slides.Add nSlide, ppLayoutBlank

    ' Lepljenje opsega u novi slajd
    pres.Slides(nSlide).Shapes.PasteSpecial ppPasteOLEObject

    ' Promena razmere zalepljenog objektya u PowerPointu
    pres.Slides(nSlide).Shapes(1).ScaleHeight dScaleFactor, msoTrue
    pres.Slides(nSlide).Shapes(1).ScaleWidth dScaleFactor, msoTrue
```

```
    ' Horizontalno i vertikalno centriranje
    ' Ne bi bilo lose da ovaj segment izvucete iz ove procedure
    ' kako biste ostvarili bolju kontrolu nad njegovim izvrsavanjem
    CenterVertically pres.Slides(nSlide), _
        pres.Slides(nSlide).Shapes(1)
    CenterHorizontally pres.Slides(nSlide), _
        pres.Slides(nSlide).Shapes(1)

End Sub

Private Sub CopyChart(pres As PowerPoint.Presentation, _
    chrt As Chart, nSlide As Integer, _
    dScaleFactor As Double)

    ' kopiranje grafikona u clipboard, u formi slike
    chrt.CopyPicture xlScreen

    ' dodavanje slajda
    pres.Slides.Add nSlide, ppLayoutBlank

    ' lepljenje grafikona u PowerPoint
    pres.Slides(nSlide).Shapes.PasteSpecial ppPasteDefault

    ' promena razmere slike
    pres.Slides(nSlide).Shapes(1).ScaleHeight dScaleFactor, msoTrue
    pres.Slides(nSlide).Shapes(1).ScaleWidth dScaleFactor, msoTrue

    ' Horizontalno i vertikalno centriranje
    ' Ne bi bilo lose da ovaj segment izvucete iz ove procedure
    ' kako biste ostvarili bolju kontrolu nad njegovim izvrsavanjem
    CenterVertically pres.Slides(nSlide), _
        pres.Slides(nSlide).Shapes(1)
    CenterHorizontally pres.Slides(nSlide), _
        pres.Slides(nSlide).Shapes(1)

End Sub

Private Function GetSaveAsName(sTitle As String) As String

    Dim sFilter As String

    sFilter = "Presentation (*.ppt), *.ppt"

    GetSaveAsName = _
        Application.GetSaveAsFilename(filefilter:=sFilter, _
        Title:=sTitle)
End Function

Private Sub CenterVertically(sl As PowerPoint.Slide, _
    sh As PowerPoint.Shape)
```

```
    Dim lHeight As Long

    lHeight = sl.Parent.PageSetup.SlideHeight
    sh.Top = (lHeight - sh.Height) / 2
End Sub

Private Sub CenterHorizontally(sl As PowerPoint.Slide, _
    sh As PowerPoint.Shape)

    Dim lWidth As Long

    lWidth = sl.Parent.PageSetup.SlideWidth
    sh.Left = (lWidth - sh.Width) / 2
End Sub
```

CreatePresentation je najvažnija procedura u ovom listingu. Već na samom početku, po načinu na koji je izvršeno deklarisanje varijabli, možete zaključiti da je u ovoj proceduri upotrebljeno rano spajanje. Procedura CreatePresentation koristi dve varijable koje reprezentuju po jedan PowerPoint objekat. Prva od njih, varijabla pod nazivom ppt, predstavlja PowerPoint.Application objekat. Poput Application objekta iz Excelovog objektnog modela, objekat PowerPoint.Application pripada samom vrhu PowerPointovog objektnog modela. Druga varijabla nosi naziv pres i njome je predstavljen PowerPointov objekat Presentation. Moglo bi se reći da je objekat Presentation otprilike ekvivalentan Workbook objektu u Excelu.

Nakon deklarisanja varijabli, neophodno je uključiti neku vrstu mehanizma za upravljanje greškama. U programskom kôdu koji je namenjen automatizaciji, suočićete se sa povećanom verovatnoćom pojave grešaka. U ovom konkretnom primeru, pobrinuo sam se jedino da uhvatim grešku, prikažem odgovarajuću poruku na ekranu, i na kraju pokrenem izvršavanje kôda za čišćenje, koji je označen labelom ExitPoint.

Nakon postavljanja reference ka radnom listu Reports, vreme je za kreiranje nove instance PowerPointa i usmeravanje ppt varijable ka toj instanci. Zatim sam pokrenuo PowerPoint i u njemu kreirao novu prezentaciju, upotrebivši uzorak pod nazivom "fall inspiring maple" ("inspirišuća jesenja javorovina", što je, složićete sasvim prikladna pozadina za mesečni izveštaj za oktobar), i novoj prezentaciji dodao naslovni slajd.

U ovom trenutku, može započeti kopiranje podataka iz Excela. Kako sam morao da obavim još gomilu stvari prilikom kopiranja iz Excelu u PowerPoint, smatrao sam da nije loša ideja kreirati posebnu proceduru koja će izvršiti ovaj zadatak. Tako je nastala procedura pod nazivom CopyDataRange. Sam proces kopiranja iz Excela u PowerPoint, obavljaju sledeće dve izjave:

```
' Kopiranje opsega u clipboard
rg.Copy
…
' Lepljenje opsega u novi slajd
pres.Slides(nSlide).Shapes.PasteSpecial ppPasteOLEObject
```

Prvi korak se, dakle, sastoji u kopiranju opsega iz Excela u clipboard. Završni deo ovog transfera obavlja se u PowerPointu, upotrebom PasteSpecial metoda objekta Shapes, uz definisanje OLEObject opcije. Ostatak ove procedure bavi se dimenzionisanjem i pozicioniranjem (uz pozivanje CenterVertically and CenterHorizontally procedura). Kako biste ovu proceduru učinili što uopštenijom (i tako povećali šanse za njenu ponovnu upotrebu), ne bi bilo loše da, čisto vežbe radi, iz ove

procedure uklonite izjave koje služe za pozicioniranje, ili da joj dodate parametar koji će vam omogućiti da kopiranje opsega na slajd izvršite bez re-pozicioniranja.

Procedura CopyChart je veoma slična CopyDataRange proceduri. Prva razlika je u tome što ova procedura zahteva Chart objekat kao parametar, a ne objekat tipa Range. Druga razlika se odnosi na operaciju kopiranja/lepljenja:

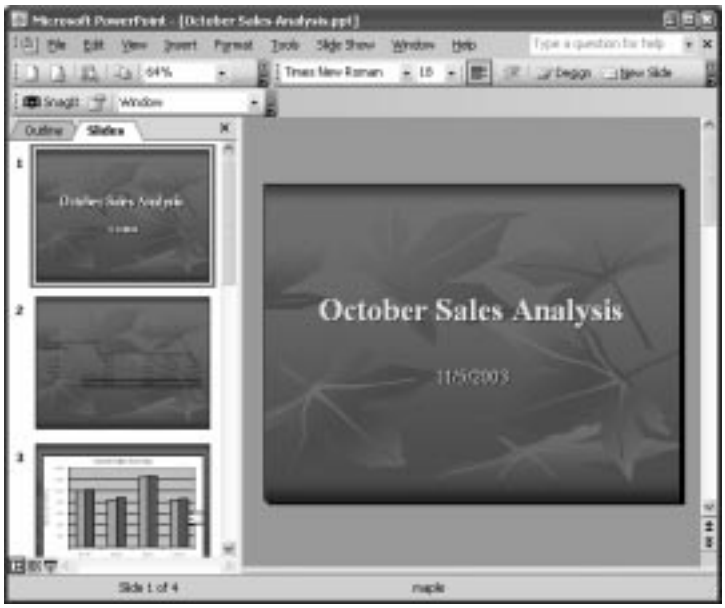
```
' kopiranje grafikona u clipboard, u formi slike
chrt.CopyPicture xlScreen
...
' lepljenje grafikona u PowerPoint
pres.Slides(nSlide).Shapes.PasteSpecial ppPasteDefault
```

Kao što vidite, upotrebio sam CopyPicture metod objekta Chart, koji vrši kopiranje slike grafikona u clipboard. Umesto upotrebe OLEObject opcije, lepljenje grafikona sam izvršio uz pomoć podrazumevane (Default) opcije.

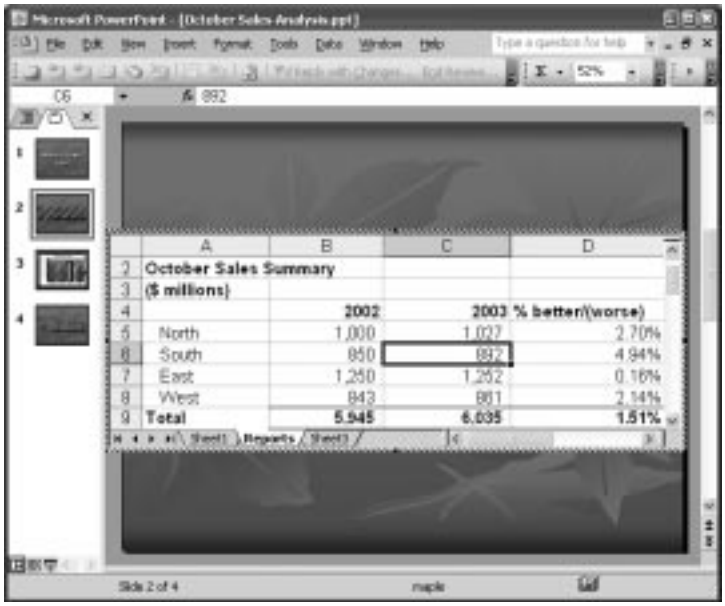
Rezultat

Listing 14.3 će, kao svoj izlazni rezultat, generisati jednu novu PowerPoint prezentaciju, sa naslovnim slajdom i tri dodatna slajda koji odgovaraju trima izveštajima iz Excela. Na slici 14.11 prikazano je kako ovi slajdovi izgledaju u PowerPointu.

Dok je grafikon ugnežđen u prezentaciju u formi slike, Sales Summary i Top 5 Stores izveštaji su ugnežđeni kao Excel dokumenti. To znači da ćete moći da iskoristite sve prednosti njihovog "editovanja na licu mesta", direktno u PowerPointu, kao što je to ilustrovano na slici 14.12.



SLIKA 14.11 *Ova prezentacija je generisana uz pomoć listinga 14.3.*



SLIKA 14.12
Ukoliko to želite, Excelove izveštaje možete editovati direktno u PowerPointu.

Mada se ovde radi o sasvim rudimentarnom primeru, imajući u vidu obilje mogućnosti koje nude PowerPoint prezentacije, nije neophodno biti genijalac da biste shvatili na koje sve načine možete upotrebiti automatizaciju, radi kreiranja izveštaja i odgovarajućih prezentacija u cikličnim vremenskim intervalima.

Zaključak

Tehnologija povezivanja i gnežđenja objekata predstavlja sjajan način da, u slučajevima kada je to opravdano, iskoristite svu snagu različitih aplikacija iz paketa Microsoft Office, istovremeno zadržavajući mogućnost da sve ove različite komponente objedinite u jedan jedinstveni dokument. Svaki dokument koji u sebi sadrži komponente kreirane uz pomoć različitih aplikacija naziva se složenim (compound) dokumentom.

Povezivanje i gnežđenje objekata je moguće prvenstveno zahvaljujući moćnoj arhitekturi koja je poznata pod nazivom Component Object Model (COM). Tokom proteklih godina, Microsoft je svoje korisnike neprestano zbunjivao čestim promenama naziva i izmenama definicije različitih COM tehnologija - naročito kada je u pitanju funkcionalnost vezana za linkovanje i gnežđenje objekata (OLE).

Iako se OLE može korisno upotrebiti u mnogim situacijama, nesumnjivo ćete se susretati i sa situacijama u kojima OLE ne predstavlja najprikladnije rešenje. U takvim slučajevima, možda će biti potrebno da primenite koncept za čije se označavanje najčešće koristi termin "automatizacija" (automation). Automatizacija se sastoji u kontrolisanju jedne aplikacije, upotrebom programskog kôda koji "živi" u nekoj drugoj aplikaciji. Primera radi, sasvim je moguće kreirati jednu Microsoft Word aplikaciju koja bi bila u potpunosti implementirana u VBA kôd koji se izvršava iz Excela (naravno, to ne znači da će vam jedna ovako besmislena ideja ikada pasti na pamet). Pri projektovanju programskih rešenja koja zahtevaju upotrebu više različitih Microsoft Office aplikacija, obično će jedna od aplikacija (Access, Excel, Word, itd.) isplivati na površinu kao dominantna aplikacija za neko konkretno rešenje, i upravo će u toj dominantnoj aplikaciji najčešće biti smešten programski kôd koji sadrži implementaciju datog rešenja.

Jedan od ključnih koncepata automatizacije jeste koncept takozvanog spajanja (binding). Spajanje, u stvari, predstavlja proces "upoznavanja" neke eksterne biblioteke klasa (poput one koja pripada Wordu), sa okruženjem u kome se dati programski kôd izvršava. Postoje dve kategorije spajanja. Rano spojeni objekti su oni objekti koji su spojeni još u fazi projektovanja, postavljanjem reference ka biblioteci klase tog objekta u VBE editoru. Kasno spojeni objekti su objekti koji se spajaju u fazi izvršavanja programa, eksplicitnim kreiranjem instance objekta uz pomoć CreateObject metoda. Zbog znatno boljih performansi, rano spajanje predstavlja najuobičajeniji način spajanja eksternih biblioteka klasa sa VBA projektima.

OLE i automatizacija se mogu veoma prikladno upotrebiti za pristup podacima koji ugnežđeni unutar drugih Microsoft Office aplikacija (kao i unutar svih ostalih aplikacija koje podržavaju OLE tehnologiju i automatizaciju). Kompjuterski svet je, međutim, mnogo, mnogo veći od podataka ugnežđenih u ove aplikacije. Radi istraživanja ogromne hrpe podataka u ovom virtualnom univerzumu, moraćete da pronađete njihov zajednički sadržalac. Važna uloga zajedničkog sadržaoca, već po tradiciji je poverena jednostavnom tekstualnom fajlu. U narednom poglavlju pozabavićemo se problemima vezanim za kreiranje i upotrebu tekstualnih fajlova.